Building a Dependency Network for Teaching-Learning of Conceptual Structures

Nagarjuna G., Meena Kharatmal, Rajiv Nair
{nagarjuna,meena,rajiv}@hbcse.tifr.res.in

Homi Bhabha Centre for Science Education, TIFR, Mumbai, India

Abstract. We propose in this paper a simple method to construct a machine processable semantic network, called a dependency network, that gathers all the concepts and skills as nodes and the relation type "depends on", (and its inverse "required for"), as their edges. As a conceptual structure can be used to compute a roadmap of any learning-teaching objective. As a prelude we build and contribute a seed graph for demonstration and introduce a collaborative portal for contributing, publishing and dynamically building dependency networks. A possible generalization of this methodology for knowledge organization is discussed.

1 Introduction

Every good author of a text book informs the reader the prior knowledge before introducing any new topic. Most of the good text books therefore explicitly mention the prerequisites at the beginning of each chapter or teachers provide a refresher course before commencing a new topic. This is due to the obvious assumption that if the prerequisites are not satisfied adequately the learner may not be able to comprehend the new ideas introduced in the book. This has been the generic guiding principle of curriculum design. This principle also stands out as one of the consensus from the widely accepted constructivist philosophy of education[1,2]. Constantly helping and reinforcing the prior knowledge to learn something new is a time-tested ancient wisdom shared among most educationists. Based on this assumption we propose a simple method of processing prerequisites for conceptual structures by employing a conceptual structure itself.

Mastering conceptual structures is a skill that requires inter-disciplinary understanding involving certain topics from domains such as logic, linguistics, mathematics, AI, databases, philosophy, computer science etc. We propose in this paper a simple method to construct a machine processable semantic network, which may be called a *dependency network*, that gathers all the concepts and skills as *nodes* and the relation type "depends on", (and its inverse "required for"), as their *edges*.

There are several studies on the dependency relation in different contexts.[3,4,5,6] Semantics and logic of dependency are covered in [6]. Keller identified some kinds of dependencies in the context of requirements analysis[4]. Cox et.al. did a more

general modeling of dependencies and an ontology of dependency relation[5]. In this paper we focus on a specific kind of dependency, which can be classified as a species of causal dependency in the context of semantics.

Arguably one of the largest operating system stack available today is the Debian GNU/Linux[1], which maintains its stable opearation by asserting dependency relations among them. Each package contains in its metadata all other packages that are required, and some of the required packages again may inturn depend on other base packages. This is possibly the largest working example where dependency network is used as a semantic structure, with more than 20,000 packages with mutual dependencies. We harvested all the asserted relations from their packages and created a dependency network, which is also available from the portal www.gnowledge.org/search_debmap?val=1, where one can query for any package and obtain the dependency graph. We further interpreted this semantic network as a complex system, because it exhibits same characters of any natural networks that have been studied.[7]. The most notable being the scale-free character and power law distribution[8].

Inspired by the way the Debian OS uses dependency relation, we think we can construct collaboratively a similar knowledge base that can give us the roadmap of any given learning objective. An artificially constructed operating system does not work if the dependencies are not met, similarly cognitive agents will not understand the meaning of a concept *explicitly* unless the prior meanings are already understood. This is therefore a kind of critical dependency[4]. While this being the case from a learner's point of view, we can make a corollary statement from a teacher's point of view: a teacher could not make a student learn unless she ensures that the prerequisites are already introduced to the student. Thus, teaching and learning requirements are linked, though the processes may not happen concurrently and to the same agents.

Based on the assumption, that prior-knowledge is essential for understanding new concepts, we undertook the program of collaboratively constructing a comprehensive propositional semantic network that uses dependency as a primary relation.

2 The Proposal

A prerequisite can be expressed as a dependency link between two concepts/skills. For example, to understand the meaning of the concept "semantic network", we expect the learner must already know the concepts "node", "links", "proposition" etc. And to understand what a proposition is we expect the learner must already know what a sentence is, a statement is, the distinction between a sentence and a proposition, to know "subject" and "predicate" the learner must already know what a sentence is etc. This prior knowledge can be mapped as a simple graph shown in Figure 1. The figure shows teaching-learning sequences for the learning objective semantic network. One may extend it by further asserting that semantic network is required for semnatic web, etc.

[1] The Debian GNU/Linux Operating System, http://www.debian.org/

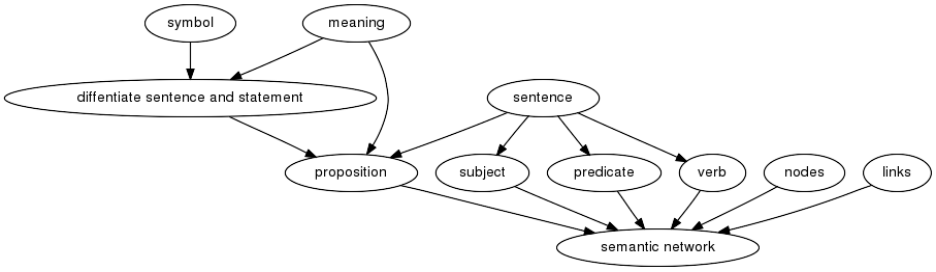

Fig. 1. A sample dependency network. The learning objective in this map is "semantic network".

All definitions will yeild a clear set of dependency assertions between *definiendum* and *definiens*. Any definiendum semantically depends on the definiens in a definition, or conversely the definiens are required for understanding the definiendum. Such dependencies may give us a conceptual network among entirely explicit and declarative knowledge. However we cannot ground all concepts in the world of concepts alone. Some concepts may need experience, an activity, a skill, solving a problem or such real experience situations that help us learn. Similarly some skill/activities may need conceptual skills, such as reading an algebraic expression. The resulting network may not contain exclusively all concepts, therefore the network may not be a strict conceptual structure. It is a conceptual/cognitive structure. Nevertheless, grammar of this network, and its computability properties, will not be different from that of a directed graph.

The resulting network, which is a directed graph, could provide the following for supporting teaching-learning and assessment process:

- **a road map** to reach a given learning objective;
- **a road-ahead map** informing what learning paths lie ahead of a given learning objective;
- **a navigational aid** providing immediate guide to inform what are next and previous steps to take direct result of being a road map);
- a dynamically generated **teaching-learning sequence** suggesting the order of presentation in, for example, a text book;
- **a relative depth measure** informing the number of levels to follow depending on the current position;
- **a surface map** of the knowledge with some special emerging properties (discussed below);
- **a directed acyclic graph**[2], which can be used in decision algorithms by an automated LMS (learning management system).

[2] Some of the nodes may have mutual dependencies. For example, concept required for class and class is required for concept. By merging all such mutual dependencies into a single node, we can construe the resulting network as a causal Bayesian kind.

For the subject of conceptual structures we collected some assertions based on the concepts introduced in the introductory chapters from [9,10,11,12]. A sample collection of assertions are as follows:

```
digraph G {
        subtype -> species;
        categories -> metatypes;
        classification -> taxonomy;
        types -> class;
        type -> supertype;
        type -> subtype;
        type -> universals;
        general -> "common nouns";
        ...
        "predicate calculus" -> "formal semantics";
        "knowledge organization" -> ontology;
        "semantic network" -> "semantic web";
        RDF -> "semantic web";
        ...
}
```

where "$\rightarrow$" stands for the inverse of the dependency relation, *required for*. A comprehensive set of such assertions as an input file can be prepared and submitted to a graph processor (in our case Graphviz[13]) to generate the directed graph. To reduce the clutter of the large graphs as well as suggesting the roadmap we could use transitive reduction filter (also available with the Graphviz library), to eliminate redundant edges that could be deduced by applying transitivity rule.[14]

To facilitate collaborative contribution of the assertions, we have created a community portal www.gnowledge.org[3] which provides the following features:

- a facility to upload the input files that contain assertions in a specified format;
- an algorithm to check preexisting nodes and edges in the memory;
- a search interface;
- dynamic generation of three different directed graphs (road-map, road-ahead-map, and a combination of the two previous graphs);
- a basic facility to edit and delete links;
- a version control feature to record who did what and when;
- an option to save the generated graphs in several image formats including SVG (scalable vector graphics);

The portal, though requires more useful features and better user interface for convenient use, we introduce this effort here to obtain the feedback from the peer group to first of all comment on its projected use and implications.

[3] Gnowledge portal http://www.gnowledge.org/.

The proposed method is general, and not specific to mapping the knowledge of conceptual structures. However, since the method used is an application of conceptual structures, and can also serve the purpose of dynamically generate the teaching learning sequences, we think it could be useful for knowledge representation community. A general introductory paper written for non-formal audience, discussing the motivation, method and possible application and impact of this method are presented in [15].

3 Observations

The sample merged graph of dependencies of the seed content cannot fit in a paper format, therefore it is uploaded at the portal in a SVG format, and also in a PDF format[4] The network contains more than 500 nodes, therefore to read the details we need to zoom in 200-300% to read the text. Due to several edges the graph looks quite cluttered, and so not very convenient to read. To extract the specific dependencies one may query from the portal and see the generated graph in isolation for a given node. Though cluttered, the merged graph has some properties worth mentioning.

Most notable observation is the nodes on the top of the graph are to be learnt or taught first, and nodes at the bottom of the graph are to be learnt later. The nodes that are aligned at the top are: and, or, sentence, object, class, words, brackets, aspect, symbols, variable, concept, types, subtypes etc. The nodes that got aligned at the bottom are: canonical graphs, specialiation rule, conceptual graphs, ontology, OWL, semantic web, conceptual model, schema, syllogism, formal semantics, proof, etc. When advanced chapters from the text books are processed, other deeper and difficult concepts and methods could appear at the bottom. This clearly indicates that using this simple method, we can frame a specific curriculum sequence more objectively.

Nodes with a large number of outgoing links cannot be neglected in education. At any given level, the first priority can be given to the nodes with larger number of outgoing links. Since, outgoing links are required for learning several other objectives/sequences, these nodes have greater potential to serve in future learning and can also be marked as core concepts. In the seed graph the nodes "class" and "relation" have high number of outgoing links. After additional assertions we may find other core concepts emerging. This needs more work.

The directed graph algorithm automatically positions the nodes and draws the edges. When merged as a large graph, unlike the isolated dependency graphs obtained by querying for one of the learning objectives, we notice that those nodes which are at a same level (a sort of latitude of the graph) are placed horizontally. When new assertions are inserted that may have links with existing nodes, the graph *accommodates* itself by *assimilating* the new nodes automatically following the well known digraph construction rules. All the adjacent nodes

[4] The hyperlinks are http://www.gnowledge.org/cs-merged-graph.svg and http://www.gnowledge.org/cs-merged-graph.pdf respectively.

descending from a parent share similar learning sequence. When a child node has more than one parent node, it indicates as many sequences to cover.

The generated directed graph is a horizontal spread with only a depth of 13 levels. As the graph indicates, the prerequisites for graph based knowledge representation are rich and belong to different disciplines. We have barely scratched the surface of the required prerequisites since we have not inserted several other assertions. For example, when the skills and conceptual requirements from linguistics, logic, graph theory, algebra, predicate calculus, computer science and philosophy are comprehensively contributed the generated graph will appear richer than the shown map. We expect the depth to grow when we insert more assertions into the knowledge base.

This data cannot be finite but can only be judgeable as comprehensive enough by the peer group. The result obtained is far from satisfactory, but we hope it can take a good shape if the community finds it necessary and begins to contribute collaboratively.

4 Discussion

Since the method used is simple and general and could be used for all domains of knowledge[5], the network, we hope, may evolve as a general mapping and sequencing tool for learning and teaching. Using the scaffolding of dependency one can add some additional tags, say modalities to code the strength of the assertions as possible or necessary. For example, while adding an exercise as a node we may add the *possibility* tag to the node since there can be other possible exercises that may give similar experience, difficulty and understanding. When the relation is core semantic dependency, as in the case of definiendum and definiens relation, one may tag with *necessity*. Additional enrichment of this network can be done by also inserting assessment objects as nodes linked to some of the learning objectives, which can be used as pre/post-test resources. Such a knowledge base could be useful for automated assessment and delivery of lessons.

Such a map, if available as a easily queryable knowledge base, a teacher or a student in a classroom can bring on board explicitly the prerequisites, so that the focus of deliberations in the class room fall on satisfying all the requirements. However, if a map of this kind is available to normal teachers as well as slow learners could do better in teaching and learning respectively, based on the assumption that explicit knowledge helps in mastering any domain. Moreover, even an artificial tutoring system can guide the learner by presenting the requirements in the order suggested by the generated roadmap and can present the assessment items at appropriate places. However these are more or less obvious applications of the map. We think the map does more than the obvious.

The students and teachers of CS can study the properties of directed graphs using this map. Since there does not exist a comprehensive repository of depen-

[5] The gnowledge.org portal is open for contributions from all domains of knowledge.

dencies of various domains of knowledge, one can meanwhile study the software dependency network such as the Debian dependency network, which exists for six major releases[6] We conducted on such study where the structure and dynamics of software dependencies as a semantic system[7]. A software package as a node in dependency network is very gross. We can undertake to map dependencies at a granular level, such as dependencies between libraries within packages, and in turn at a much more granular level between the functions within the libraries. This is a massive task, but doable since every dependency is explicitly asserted in a software. The students could write programs/algorithms that will extract the dependencies at various levels of granularity. We think such a massive network can become a basis for conducting several other serious semantic and computing studies.

Returning to the natural knowledge network context, the CS students could write decision algorithms for an artificial tutoring system based on dependency networks agregating at gnowledge.org: to locate learning objectives that are core (higher number of outgoing links); locate cognitively interesting terminal destinations (nodes with higher number of incoming links); calculate the roadmap of a given learning objective from a relative starting node; generate pre-test and post-test sets for administering assessments; a method for generating clusters that have similar but not same dependencies; measuring the semantic distance between nodes using dependency; write web services for accessing this information from a server to a remote client; etc.

A computer processable dependency network can also serve as a scaffolding for *parking* ontologies and other conceptual structures. This is possible since every conceivable node of any ontology or any conceptual structure can also be part of the dependency network. No formal proof for this claim is offered here, but an argument can be as follows. Given the holistic assumption that there are no loosely hanging concepts in a knowledge network[16], every concept will have a prerequisite network of nodes. This also follows from the need of at least two concepts to explicitly individuate a concept.[17] Currently all ontologies are held together by an artificial *summum bonum*, the "thing" super-class. If we hold the scaffolding of dependency network as a surface map, which is constructed using semantic flow relationships, a *natural* way of holding together all other ontologies seems possible. Could this be the topography of knowledge?

Since each node in this network has only a set of incoming and outgoing nodes, each node must have a unique set of nieghbouring nodes when all relations are fully specified. Can this criteria of unique meaning be demonstrated practically? If so, this could become a criterion for disambiguation. If the above is possible, we can obtain a more realistic measure for semantic distance based on the obtained directed graph. Given any two nodes in the dependency network, we can compute the order of separation between the nodes.

Based on the reasons and speculated implications of the program, we hope the dependency network of any knowledge can be constructed. Since, the community of conceptual structures, semantic web, AI, expert systems, cognitive scientists

[6] Starting from 1994 Debian released six major releases.

have the interest to map knowledge in general, this may become a useful resource for both research and material development.

To sum up, we proposed a method of constructing a dependency network for the domain of conceptual structures, and argued that it can be a general strategy for any other domain.

References

1. Ausubel, D., Novak, J., Hanesian, H.: Cognitive Physchology: A Cognitive View. Holt, Rinehart and Winston, New York (1978)
2. Ausubel, D.: The Psychology of Meaningful Verbal Learning. Grune & Stratton, Oxford, England (1963)
3. Keller, A., Kar, G.: Dynamic dependencies in application service management. In: 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), Citeseer (2000)
4. Keller, A., Blumenthal, U., Kar, G.: Classification and computation of dependencies for distributed management. In: Proceedings of the Fifth IEEE Symposium on Computers and Communications (ISCC 2000), Citeseer (2000) 78
5. Cox, L., Delugach, H., Skipper, D.: Dependency analysis using conceptual graphs. In: Proceedings of the 9th International Conference on Conceptual Structures, ICCS, Citeseer (2001)
6. Pearl, J., Verma, T.: The logic of representing dependencies by directed graphs (1987)
7. Ray, A., Nair, R., Nagarjuna, G.: Saturation in the scale-free dependency networks of free and open-source software. Arxiv preprint arXiv:0901.4904 (2009)
8. Zipf, G.K.: Human Behavior and the Principle of Least Effort. Addison-Wesley, Cambridge, Massachusetts (1949)
9. Sowa, J.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley Publishing Company, USA (1984)
10. Sowa, J.: Knowledge Representation: Logical, Philosophical and Computational Foundations. Brooks/Cole, USA (2003)
11. Chein, M., Mugnier, M.: Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs. Springer-Verlag New York Inc (2008)
12. Sowa, J.F.: Semantic networks. In Shapiro, S.C., ed.: Encyclopedia of Artificial Intelligence. 2 edn. John Wiley & Sons, Inc., New York (1992)
13. Ellson, J., Gansner, E., Koutsofios, L., North, S., Woodhull, G.: Graphviz-open source graph drawing tools. Lecture Notes in Computer Science (2002) 483–484
14. Aho, A.V., Garey, M.R., Ullman, J.D.: The transitive reduction of a directed graph. SIAM Journal on Computing **1** (1972) 131–137
15. Nagarjuna G.: Collaborative creation of teaching learning sequences and an atlas of knowledgge. Mathematics Teaching-Research Journal Online **3** (2009) 23–40
16. Quine, W.: From A Logical Point Of View. Harward University Press, Massachusetts (1953)
17. Strawson, P.: Individuals: An essay in descriptive metaphysics. Methuen London (1971)